Imports

```
!  pip  install  networkx
!  pip  install  plotly
!  pip  install  colorlover
```

```
Requirement already satisfied: networkx in /usr/local/lib/python3.6/dist-packages (2.4)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (4.1.1)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from plotly)
Requirement already satisfied: colorlover in /usr/local/lib/python3.6/dist-packages (0.3.0)
```

```
import  networkx  as  nx
import  pandas  as  pd
import  matplotlib.pyplot  as  plt
import  colorlover  as  cl
from  IPython.display  import  HTML
import  random
from  plotly.offline  import  download_plotlyjs,  init_notebook_mode,  plot,  iplot
from  plotly.graph_objs  import  *
import  plotly.graph_objects  as  go
init_notebook_mode(connected=True)
import  nltk
from  nltk.corpus  import  stopwords
nltk.download("stopwords")
from  collections  import  Counter
```

Load Data

```
df  =  pd.read_csv("tweets2009-06-0115.csv.zip",  sep='\t',  compression='zip')
```

Q1. Choose a hash-tag

```
tehranTag  =  df[df["tweet"].str.lower().str.contains("#tehran",  na=False)].copy()
tehranTag.head()
```

Q2. Build a Mention Graph

```
def addMentionedColumn(df):

    def mentionsList(txt):
        allWords = [word.strip("""  ,.:'\";""").lower() for word in txt.split()]
        allNames = [word.strip("@") for word in allWords if word.startswith("@")]
        uniqueNames = list(set(allNames))
        return allNames

    df["mentioned"] = df["tweet"].apply(mentionsList)
```

```
def mentionGraph(df):
    g = nx.Graph()

    for (index, date, user, tweet, mentionedUsers) in df.itertuples():
        for mentionedUser in mentionedUsers:
            if (user in g) and (mentionedUser in g[user]):
                g[user][mentionedUser]["numberMentions"] += 1
            else:
                g.add_edge(user, mentionedUser, numberMentions=1)

    return g
```

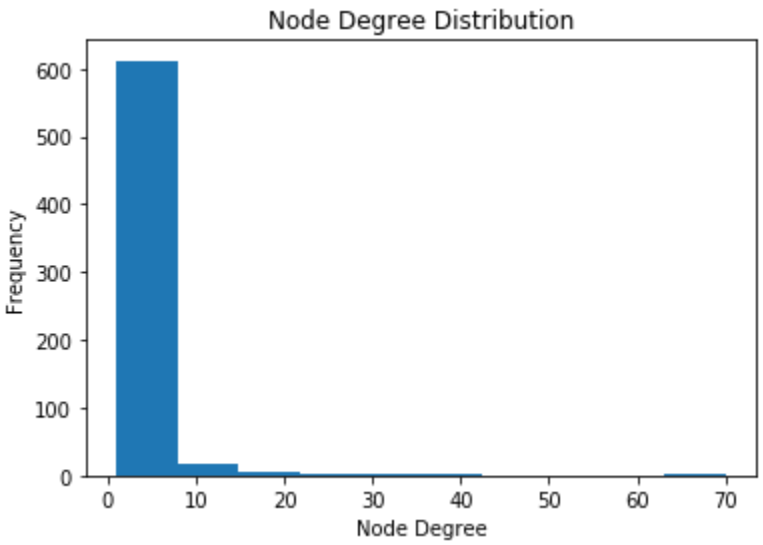
```
# Prepare dataset
addMentionedColumn(tehranTag)

# Create mention graph
tagGraph = mentionGraph(tehranTag)
```

(a) How many nodes and how many edges in your mention graph?

```
print("# nodes:", len(tagGraph.nodes()))
print("# edges:", len(tagGraph.edges()))
```

(b) Build a histogram of the graph nodes' degree

```
plt.hist(list(dict(nx.degree(tagGraph)).values()))
plt.title("Node  Degree  Distribution")
plt.xlabel("Node  Degree")
plt.ylabel("Frequency")
fig  =  plt.gcf()
```



Only few accounts are mentioned by many people.

(c) Provide a list of top 5 edges with highest weights

```
def  countEdgeWeight(graph):
        edgeList  =  []
        for  node1,  node2  in  graph.edges():
                edgeList.append([node1,  node2,  tagGraph[node1][node2]['numberMentions']])

        weightDf  =  pd.DataFrame(edgeList)
        weightDf.columns  =  ["Node  1",  "Node  2",  "Edge  Weight"]
        return  weightDf
```

```
#    Find  top  5  edges  by  weight
edgeWeightList  =  countEdgeWeight(tagGraph)
edgeWeightList.sort_values('Edge  Weight',ascending=0).head(5)
```

	Node 1	Node 2	Edge Weight
91	anabell39	mousavi1388	4
385	gita	grahattalab	4
197	steve_schippert	resemblance	3
295	gr8rdh	mommadona	3
75	jslefanu	potent_one	3

```
tagGraph.edges.data()
```

```
EdgeDataView([('danieldoyle', 'breakingtweets', {'numberMentions': 2}), ('breakingtweets',
```

(d) Provide a visualization of the mention graph in which the edge color reflects its w

```
def configure_plotly_browser_state():
    import IPython
    display(IPython.core.display.HTML('''
                <script src="/static/components/requirejs/require.js"></script>
                <script>
                    requirejs.config({
                        paths: {
                            base: '/static/base',
                            plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',
                        },
                    });
                </script>
                '''))
```

```
def addRandomPositions(graph):
        posDict = dict((node,(random.gauss(0,10),random.gauss(0,10))) for node in graph.noc
        nx.set_node_attributes(graph, name="pos", values=posDict)
```

```
def getLineColor(edgeWidth):
                cells = 300
                # map color scale to edge weight scale
                blues = cl.scales['9']['seq']['PuRd']
                weightColor = cl.interp(blues, cells)
                lineColor = int((cells-1)*((edgeWidth-minEdgeWeight)/(maxEdgeWeight-minEdgeWeig

                return weightColor[lineColor]
```

```
def plotNetworkWeightColor(graph,minEdgeWeight, maxEdgeWeight):
    closenessCentr = nx.closeness_centrality(graph)
    maxCentr = max(closenessCentr.values())
    minCentr = min(closenessCentr.values())

    scatters=[]

    for (node1, node2) in graph.edges():
        x0, y0 = graph.nodes[node1]['pos']
        x1, y1 = graph.nodes[node2]['pos']
        edgeWidth = graph[node1][node2]['numberMentions']
        s = Scatter(
                    x=[x0, x1],
                    y=[y0, y1],
                    hoverinfo='none',
                    mode='lines',
                    line=scatter.Line(color=getLineColor(edgeWidth)))
        scatters.append(s)



    for node in graph.nodes():
        xPos, yPos = graph.nodes[node]['pos']
        s = Scatter(
                    x=[xPos],
                    y=[yPos],
                    hoverinfo='none',
                    mode='markers',
                    marker=dict(
                        color="#888",
                        size=10,
                        line=dict(width=2)))
        scatters.append(s)

    layout = Layout(showlegend=False)
    fig = Figure(data=scatters, layout=layout)
    iplot(fig, show_link=False)
```

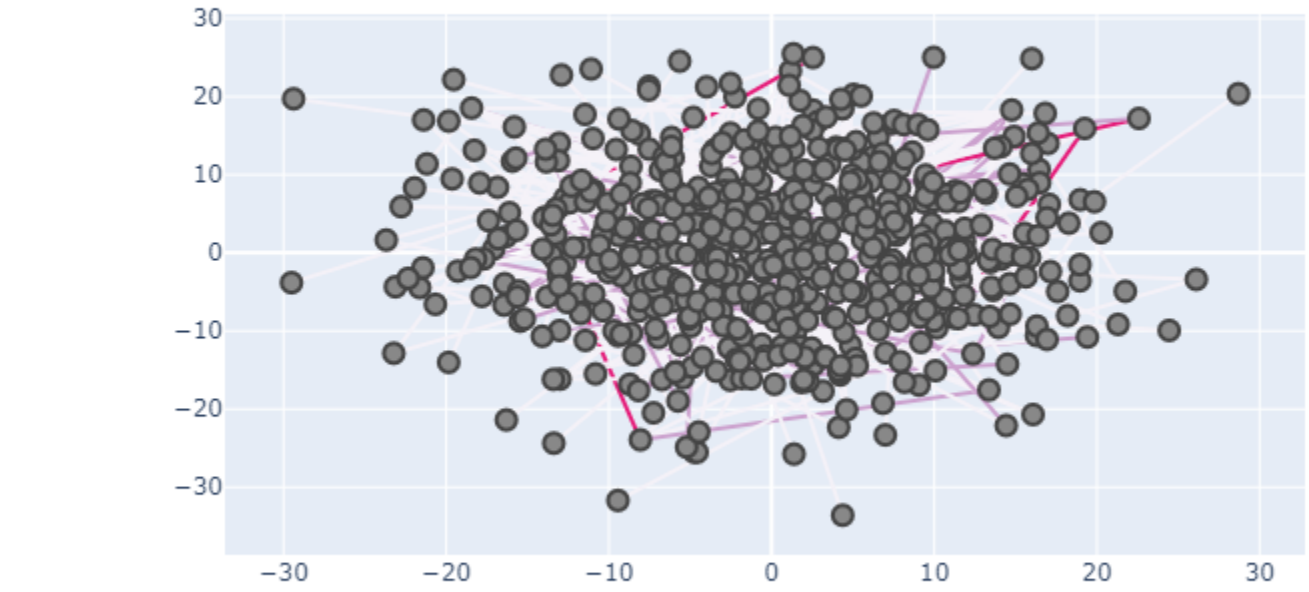
```
addRandomPositions(tagGraph)
```

```
# find max and min edge weights in the graph
maxEdgeWeight = edgeWeightList.sort_values('Edge Weight',ascending=0).head(5).as_matrix()[0][2
minEdgeWeight = edgeWeightList.sort_values('Edge Weight',ascending=1).head(5).as_matrix()[0][2
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
```

```
# Plot network where line color indicates edge weight
# configure_plotly_browser_state()
# plotNetworkWeightColor(tagGraph, minEdgeWeight, maxEdgeWeight)
```

```
display(Image('/content/drive/My Drive/newplot (1).png'))
```



Q3. Content Analysis

```
def getUserTopKWords(df, userList, k=3):
    userWords = {}
    for user in userList:
        topkWords = countWords(df.loc[df['user']==user]['tweet'].as_matrix()).most_comm
```

```
                tweetCount = len(df.loc[df['user']==user]['tweet'])

                words = {}
                for word, count in topkWords:
                        words[word] = count

                userWords[user] = {'tweetCount':tweetCount, 'words':words}

        return userWords
```

```
stopwordList = set(stopwords.words('english'))

# Add Possible Stop Words for twitter
stopwordList.add('http')
stopwordList.add('com')
def cleanword(word):
        return bool(len(word)>2 and
                            not word.startswith("@") and       # Remove users
                            not word.startswith("#") and       # Remove other hash-tags
                            not word.startswith("http") and # Remove links
                            word not in stopwordList)

# Count word frequency in a list of documents, excluding terms in a stopword list
def countWords(corpus):
        counter = Counter()
        # Open the doc in the corpus and count the word frequency
        for doc in corpus:
                allWords = [word.strip("""  ,.:-'\";""").lower() for word in doc.split()]
                counter.update([word for word in allWords if cleanword(word)])
        return counter
```

(a)Analyze the most common words in all the tweets

```
# Top words that appear across all tweets for the selected hash-tag
pd.DataFrame(countWords(tehranTag['tweet'].as_matrix()).most_common(10), columns=["Word", "Cour
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning:

Method .as_matrix will be removed in a future version. Use .values instead.
```

	Word	Count
0	iran	153
1	tehran	120
2	police	104
3	twitter	73
4	university	52
5	word	51
6	attacked	49
7	situation	48
8	photos	46
9	girls	46

Main themes of the tweets include talking about the situation of iran, and worrying about the po

(b)Plot network with top words (k=3), add hover information for the nodes

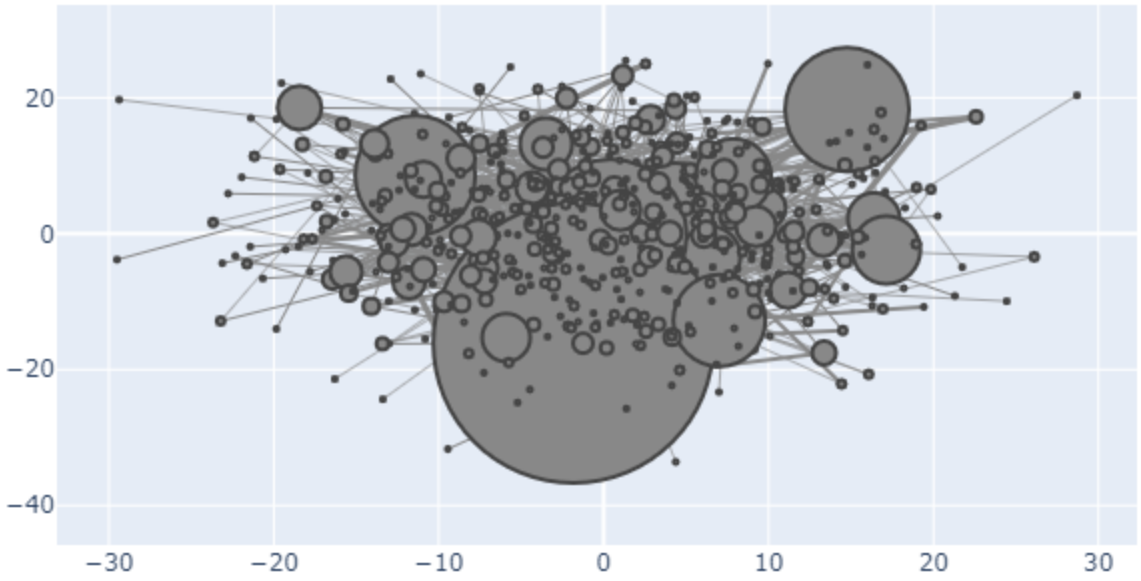
```
def plotNetworkWeightColor_Q3(graph,minEdgeWeight, maxEdgeWeight,userTopK,centr=None):
    # Format node text labels to dispaly username and most common words
    def formatLabel(username):
        numTweet = userTopK[username]['tweetCount']
        userTopWords = userTopK[username]['words']
        topWords = ''
        for word in sorted(userTopWords, key=userTopWords.__getitem__, reverse=True)
            topWords += '{0} ({1}) '.format(word, userTopWords[word])
        if centr==None:
            return "User: {0}<br>Top Words: {1}<br>Number of Tweets: {2}".format(u
        else:
            nodeCentr = centr[node]
            return "User: {0}<br>Top Words: {1}<br>Number of Tweets: {2}<br>Closer
    def getMarker(node):
        if centr == None:
            return dict(color='#888', size = nx.degree(graph,node)*2, line=dict(
        else:
            cells = 300
            # map purd color scale
            purd = cl.scales['9']['seq']['PuRd']
```

```
                    purd = cl.scales[9][seq][rgb]
                    purd300 = cl.interp(purd, cells)
                    maxCentr = max(centr.values())
                    minCentr = min(centr.values())
                    nodeCentr = centr[node]
                    nodeColor = int((cells-1)*(nodeCentr-minCentr)/(maxCentr-minCentr))

                    return dict(color=purd300[nodeColor],
                                        size =nx.degree(graph,node)*2,
                                        line=dict(width=2))
    scatters=[]

    for (node1, node2) in graph.edges():
        x0, y0 = graph.nodes[node1]['pos']
        x1, y1 = graph.nodes[node2]['pos']
        edgeWidth = graph[node1][node2]['numberMentions']
        s = Scatter(
                    x=[x0, x1],
                    y=[y0, y1],
                    hoverinfo='none',
                    mode='lines',
                    line=scatter.Line(width=edgeWidth ,color='#888'))
        scatters.append(s)



    for node in graph.nodes():
        xPos, yPos = graph.nodes[node]['pos']
        # Format node label
        labelText = 'User: {0}'.format(node)
        if bool(userTopK) and len(node)>0:
            labelText = formatLabel(node)

        s = Scatter(
                    x=[xPos],
                    y=[yPos],
                    text=labelText,
                    hoverinfo='text',
                    mode='markers',
                    marker=getMarker(node))
        scatters.append(s)
    layout = Layout(showlegend=False)
    fig = Figure(data=scatters, layout=layout)
    iplot(fig, show_link=False)
```

```
Plot network with top words (k=3), where size is number of tweets
onfigure_plotly_browser_state()
lotNetworkWeightColor_Q3(tagGraph,minEdgeWeight, maxEdgeWeight, getUserTopKWords(tehranTag,tagG
```

```
display(IPython.core.display.HTML('<a href="/content/drive/My Drive/graph.html">'))
```

```
display(Image('/content/drive/My Drive/newplot.png'))
```



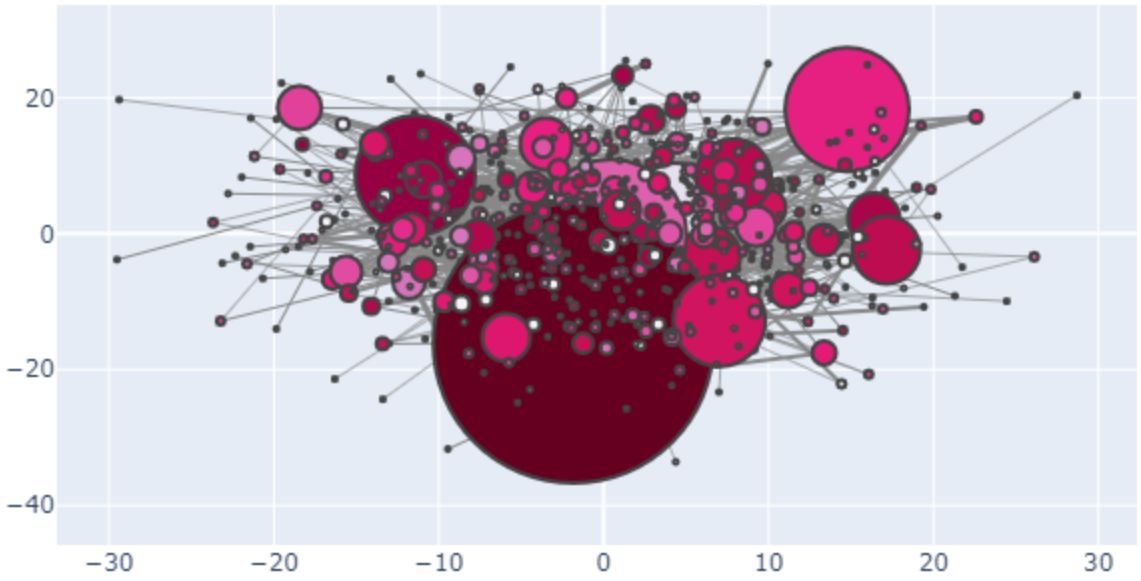
Q4. Centrality Analysis

```
nx.closeness_centrality(tagGraph)
```

```
nx.betweenness_centrality(tagGraph)
```

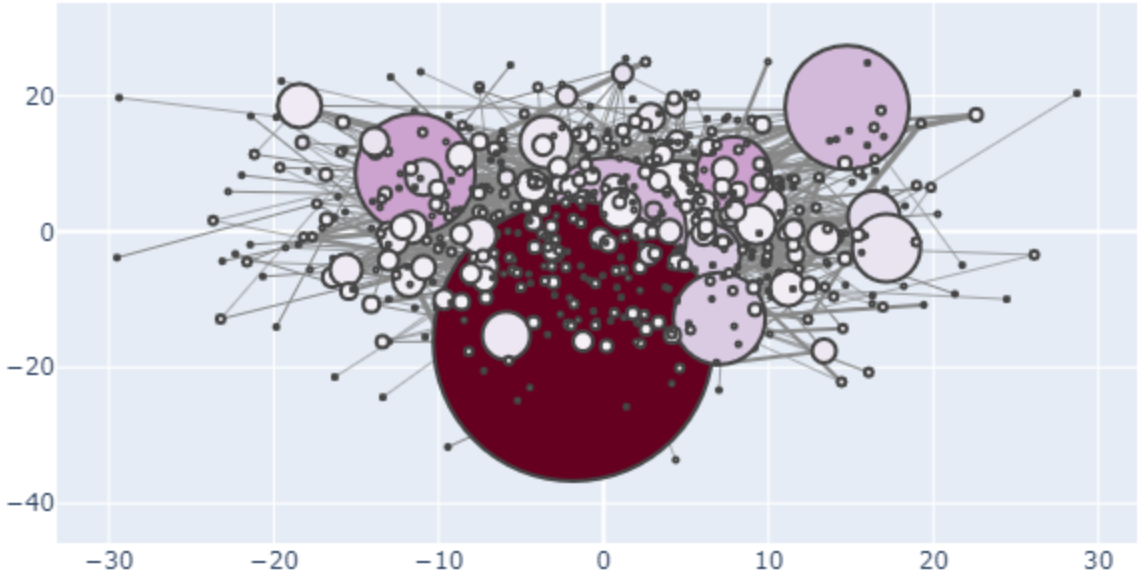
```
=3), where size is number of tweets

ninEdgeWeight, maxEdgeWeight, getUserTopKWords(tehranTag,tagGraph.nodes()),nx.closeness_central
```

```
display(Image('/content/drive/My Drive/newplot (2).png'))
```




```
k=3),  where  size  is  number  of  tweets

nEdgeWeight,  maxEdgeWeight,  getUserTopKWords(tehranTag,tagGraph.nodes()),nx.betweenness_central
```

```
display(Image('/content/drive/My  Drive/newplot  (3).png'))
```



The key players of two methods are mostly the same. But from betweenness measure graph we
the biggest key player more obviously than the closeness measure.

Of the tested methods, the betweeness measure was the most useful. The closeness measure
identify the major players with as many nodes had similar centality values.

Q5. Connectivity Patterns

(a)

```
print("Number of Maximal cliques: {0}".format(len([clique for clique in nx.find_cliques(t
```

```
Number of Maximal cliques: 569
```

```
print("Graph's clique number: {0}".format(nx.graph_clique_number(tagGraph)))
```

```
Graph's clique number: 4
```

```
# 3.) Maximal cliques for each node
pd.DataFrame(list(nx.number_of_cliques(tagGraph).items()),
                        columns=['user', 'Maximal clique']).sort_values('Maximal clique',asc
```

	user	Maximal clique
188	gita	54
81	wpxlse	38
62	iran09	24
1	breakingtweets	24
21	simonscotland	19
...	...	...
266	kimmyville	1
265	rvercesi	1
261	cnn)!?!	1
259	danweinbaum	1
636	dijitalboy	1

637 rows × 2 columns

```
# 4.) Size of the largest maximal clique containing each given node
pd.DataFrame(list(nx.node_clique_number(tagGraph).items()),
                        columns=['user', 'Maximal clique']).sort_values('Maximal clique',asc
```

	user	Maximal clique
318	solidadrocks	4
403	profbrian	4
192	gmarkham	4
194	blindcyclists	4
196	getsmartmoodle	4
...	...	...
237	iranriot	2
236	dirk2112	2
234	surya_source	2
233	andrewfynn	2
636	dijitalboy	2

637 rows × 2 columns

(b)

```
tehranTag[tehranTag['user']=='solidadrocks']
```

	date	user	tweet	
2221349	2009-06-13 18:33:36	solidadrocks	Telephone communication between Tehran and the...	
2588927	2009-06-14 02:31:02	solidadrocks	RT @schillingfan @Misha1234 @pourmecoffee: Sli...	[schil pou
2603259	2009-06-14 02:47:35	solidadrocks	RT @cbn2 @Cody_K Fatwa Issued for Changing the...	[cb
2647629	2009-06-14 03:43:05	solidadrocks	Audio: Last words of Mousavi before getting ar...	
2731118	2009-06-14 05:39:24	solidadrocks	Panic on the streets of Tehran. Outside mass p...	
2755115	2009-06-14 06:19:33	solidadrocks	RT @NewIRAN RT @Gita they've shattered everyth...	[newir

Based on the above values, this indicates that this network has a lot of cliques, but they are all f
Combined with the centrality measures in Q4, and reviewing the tweet content, this suggests tha

users are small groups of friends and concerned about the politics of iran.

```
from  google.colab  import  drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189898

Enter your authorization code:
··········
Mounted at /content/drive
```

```
from  IPython.display  import  Image,  display
```

```

```